that implements software including an individual computer such as a personal computer, mainframe computer, minicomputer, personal data assistant (PDA), etc. In addition computer system refers to any type of network of computers, such as a network of computers in a business, the Internet, computers connected by wired or wireless connections to one or more computers on either a permanent or temporary basis, etc. A computer system also refers to electronic devices that implement one or more functions of a computer such as: a cell phone, a television, a videogame console, a compressed audio or video player such as an MP3 player, a DVD player, a microwave oven, etc. The meta-implementation layer and/or computer integration engine may be stored on an individual computer or be stored on one or more computers of a network of computers

[0081] For the purposes of the present invention, the term "user" refers to all users of a meta-implementation layer, a component integration engine, or any other software program. A user may be an end user, a programmer, a system administrator, a software developer, another computer system, etc.

[0082] For the purposes of the present invention, the term "visual display device" or "visual display apparatus" includes any type of visual display device or apparatus such as a CRT monitor, LCD screen, LEDs, a projected display, a printer for printing out an image such as a picture and/or text, etc. A visual display device may be a part of another device such as a computer monitor, television, projector, telephone, laptop computer, watch, kitchen appliance, electronic organ, automatic teller machine (ATM) etc.

[0083] Description

[0084] The present invention provides a meta-implementation layer for accessing metadata is provided, which is not only ubiquitously available for every function and data structure available in a computer system but is also compiled into the program so it is faster than existing metadata methods. Metadata can be defined for any structure, allowing it to describe object-oriented and non-object oriented structures, and includes a definition of the structure, including methods, attributes, constructors, destructors, and events. Metadata is used to convey the description of data that is acceptable, filter values if they are not acceptable, get or set the current values on a specific object or data structure, and invoke execution of an operation.

[0085] The present invention provides a method of accessing objects constructors, methods, and attributes through metadata in a compiled form that does not require a specific interface, implementation or naming convention for the implementation. Metadata for an object does not require that the object implements a specific interface, extends a specific parent class, or follow a specific naming convention as is required by other contemporary metadata mechanisms. Metadata for an object does not require the object to implement the metadata mechanics. By removing metadata mechanics from the object, legacy and existing code does not have to be rewritten thereby avoiding the introduction of new errors into the code. Metadata mechanics allow compiled or dynamic access. Dynamic mechanics requiring an object implement a specific interface or naming convention and are in common use. Compiled metadata mechanics may be faster than dynamic mechanics and are not in use elsewhere contemporary to this invention.

[0086] The present invention also provides a component integration engine, which is a system and method for integrating objects and data within an object-oriented computing environment using metadata. The component integration engine manages the interactions between two or more objects and between objects and data, simplifying integration and limiting unexpected side effects. The unique combination of centralized managed resources, metadata for configuring objects and data, and a command architecture which uses metadata to describe input, instructions, and output creates a new type of software application hereafter referred to as a component integration engine or a component integration engine. The component integration engine uses the command in conjunction with metadata to combine small, simple commands into assemblies that perform complex processes and software applications. The engine does this by using metadata to setup input data, using metadata to invoke methods in a specific order, and returning the final result. Data and method invocations through metadata provide the component integration engine with the benefit of class indirection, object indirection, function substitution and data substitution.

[0087] The present invention also provides a new pattern for displaying and controlling model information is provided, which modifies the traditional "model view controller" pattern using metadata to allow for more flexibility and code reuse.

[0088] The component integration engine of the present invention provides a combination of a) centralized managed resources b) metadata for configuring all objects and structured data in the running system, and c) a command architecture which uses metadata to describe input, instructions, and output.

[0089] In embodiments of the meta-implementation layer and component integration engine of the present invention, a set of centralized managers is responsible for managing objects that describe common functionality such as access to shared resources (files, databases, email servers, caches), parallel processing (thread pools, user access points), and security (login, authorization, code validation, data validation). These managers are themselves managed by a "ManagerManager". Any manager or managed component can be interchanged with another managed component that is accepted by the manager. Programs accessing managed resources by identity have no way of knowing what specific object will be retrieved from a manager, only that it will fit the type of the object requested to perform a specific task. This decouples object connections, since objects reference the managers to access other objects rather than referencing these objects directly. This indirection allows model substitution, object substitution, functional substitution and data substitution to occur at run-time. A manager can manage any object that is side effect free.

[0090] Objects that are side effect free can be used and shared between many processes concurrently. By retrieving an object by a known identity or by a query, changes in manager configuration allow model substitution, object substitution and functional substitution at run-time.

[0091] Objects that are not side effect free can be retrieved from component selectors. A component selector is itself side effect free, so it can be managed. The selector accepts the type of object that is requested, and returns the appro-